Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleš Koncilja

Pridobivanje, integracija in prikaz podatkov za potrebe cenilcev

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Aljaž Zrnec

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pri projektu Cenilci v IT podjetju se je pojavila težava, da je trenutni vir podatkov – GURS spremenil shemo podatkov in politiko njihovega posredovanja. Naročniki – cenilci v svoj sistem pridobivajo podatke vsakih nekaj mesecev v obliki tekstovnih datotek na CD-ju, ki jih s pomočjo aplikacije uvozijo v podatkovno bazo. Projekt Cenilci je spletna aplikacija, ki je sposobna prikazovati posle nepremičnin. Uporabniki se tako srečujejo s težavo, da operirajo s starimi podatki, kar pa jih omejuje pri svojem delu. Ker je GURS je spremenil shemo podatkov, je potrebno narediti prilagoditev aplikacije na novo shemo podatkov. Zaradi nove možnosti pridobivanja podatkov o poslih preko spletnih servisov, implementirajte ta način pridobivanja podatkov. Obstoječo aplikacijo tudi nadgradite s sistemom prikaza nepremičnin v prostoru s pomočjo "Google Street View"API-ja, ki bo uporabnikom nudil tudi možnost pogleda nepremičnine v obliki slik in ne le v s pomočjo podatkov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Aleš Koncilja sem avtor diplomskega dela z naslovom:

Pridobivanje, integracija in prikaz podatkov za potrebe cenilcev

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Aljaža Zrneca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 10. septembra 2015 Podpis avtorja:

Zahvaljujem se Janji, staršem in vsem mojim bližnjim, ki so me podpirali skozi vsa leta študija. Posebno se zahvaljujem mentorju viš. pred. dr. Aljažu Zrnecu za strokovno pomoč in podporo pri izdelavi diplomske naloge.

Še posebej bi se zahvalil Mateju Čeliku, za strokovno in idejno pomoč.

Staršem.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application programming interface	Aplikacijski programski vmesnik
ETN	Real Estate Market Record	Evidenca trga nepremičnin
GURS	The Surveying and Mapping Authority of the Republic of Slovenia	Geodetska uprava Republike Slovenije
HTTP	Hypertext Transfer Protocol	Protokol za prenos hiperteksta
SOAP	Simple Object Access Protocol	Protokol za spletne storitve, ki temelji na XML
URL	Uniform Resource Locator	Internetni naslov, na katerem se nahaja vsebina
XML	Extensible Markup Language	Razširljiv označevalni jezik

Povzetek

V diplomskem delu je predstavljen opis sprememb obstoječe aplikacije Cenilci in razvoj prototipov novih funkcionalnosti. V uvodu je predstavljeno trenutno stanje aplikacije in trenutni način pridobivanja podatkov. Predstavljen je problem spremenjene sheme podatkov. Zaradi spremenjene sheme podatkov, je potrebna sprememba in prilagoditev aplikacije. V tretjem poglavju je predstavljena prilagoditev aplikacije na novo shemo podatkov in prilagoditev programa za uvoz podatkov za potrebe nove sheme podatkov. Predstavljene so tudi uporabljene tehnologije za razvoj aplikacije. V petem poglavju je podrobneje opisan postopek razvoja prototipa prikaza nepremičnin s pomočjo uličnega pogleda *Google Street View*. V zadnjem delu je opisan razvoj naprednega načina pridobivanja ETN podatkov z GURS-a prek spletnega servisa.

Ključne besede: shema podatkov, nepremičnina, Google Street View API, spletni servis, avtomatski prenos podatkov, ETN podatki.

Abstract

This diploma thesis presents a description of the changes of existing application Cenilci and prototype development of new functionalities. Current state of application and the current method of obtaining data is presented in the introduction of this document. A problem of amending scheme data is also described. Due to changes in the scheme of data, a change and adaptation of application is needed. The third chapter presents the adaptation of applications to the new scheme of data and also the adaption of the program for importing data for needs of the new scheme data. It also presents the technologies used to develop application. The fifth chapter describes in detail the process of developing a prototype of presentation of real estate with the help of *Google Street View*. The last part of this document presents the development of advanced method of obtaining EMR data with SMARS-through a web service.

Keywords: scheme of data, real estate, Google Street View API, web servis, automatic transfer of data, REM data.

Poglavje 1

Uvod

Dandanes niso nenehne spremembe pri razvoju računalniških sistemov dandanes niso nobena redkost. Spreminjanje, prilagajanje in nadgradnje sistemov postaja vse pogosteje opravilo načrtovalcev in razvijalcev računalniških aplikacij in sistemov. Načrtovanje in razvoj zahteva pri realizaciji sistemov kar nekaj časa. Zaradi nenehnega prilagajanja uporabnikom so potrebne spremembe, reorganizacija in nadgradnje sistemov z novimi funkcionalnostmi. Veliki informacijski sistemi so običajno sestavljeni iz več manjših podsistemov. Sprememba na enem podsistemu včasih povzroči težave pri delovanju ostalih delov sistema oz. drugih sistemih.

Aplikacija Cenilci je namenjena prikazovanju podatkov iz zbirke Evidence trga nepremičnin - ETN, ki je last Geodetske uprave Republike Slovenije (GURS). Zaradi spremenjene sheme podatkov ETN ni več mogoče uporabljati novih podatkov ETN v aplikaciji, saj nova shema podatkov ni podprta. Običajno je administrator sistema vsakih nekaj mesecev s posebnim programom za uvoz podatkov ETN v podatkovno bazo uvozil podatke, sedaj pa to zaradi spremenjene sheme podatkov ni več mogoče. Zaradi ključnega pomena dostopa do svežih podatkov ETN, je sistem za uporabnike vse manj uporaben.

Sprememba sheme podatkov v zbirki ETN, ki se nahaja na GURS-u, je povzročila težave v aplikaciji Cenilci. Aplikacija Cenilci prikazuje podatke

ETN, ki jih s posebnim programom skrbnik sistema uvozi v podatkovno bazo. Podatke skrbnik sistema predhodno pridobi s strani GURS-a v obliki tekstnovnih datotek.

Odločili smo se, da razvijemo prototip aplikacije Cenilci, ki bo delovala na novi shemi podatkov. Ker se zavedamo pomembnosti svežih ETN podatkov za uporabnika, bomo v okviru razvoja prototipa aplikacije implementirali tudi modul, ki omogoča pridobivanje svežih ETN podatkov. Uporabili bomo način pridobivanje podatkov preko spletnega servisa. Ker gre za aplikacijo, ki prikazuje posle nepremičnin, je zaradi boljše uporabniške izkušnje izjemnega pomena tudi grafični prikaz nepremičnin (sestavni deli poslov). Za grafični prikaz nepremičnin bomo razvili prototip funkcionalnosti prikaza nepremičnin s pomočjo *Google Street View API-ja*.

Prototip spremembe aplikacije in razvoj prototipa dodatnih funkcionalnosti aplikacije je opisan v nadaljevanju diplomske naloge. V prvem delu diplomske naloge bomo predstavili obstoječe stanje aplikacije, t.j. stanje pred prilagoditvijo in razvojem novih funkcionalnosti ter obstoječ način pridobivanja podatkov. V nadaljevanju bomo na kratko predstavili uporabljene tehnologije in programska orodja, ki smo jih uporabljali pri razvoju prototipa aplikacije. V petem delu naloge bo predstavljen postopek razvoja prikaza nepremičnin z uličnim pogledom. Šesti del naloge opisuje razvoj funkcionalnosti avtomatskega pridobivanja podatkov prek GURS spletnega servisa.

Cilj naloge je bil razviti prototip aplikacije, ki bo prikazovala podatke iz nove sheme ETN podatkov, prikaz nepremičnin z *Google Street View API-jem* in modul, ki bo avtomatsko prenašal zadnje objavljene podatke iz zbirke ETN.

Poglavje 2

Opis obstoječega stanja aplikacije

Spreminjanje in nadgradnja aplikacije zahteva poznavanje obstoječega delovanja aplikacije. Za kvalitetno načrtovanje sprememb in razvoj novih funkcionalnosti je potreben natančen pregled obstoječega stanja aplikacije. Izdelali smo podrobno analizo delov aplikacije na katerih bodo potrebne spremembe in nadgradnje. Analizirali smo obstoječo shemo podatkov, način prikaza nepremičnin in način pridobivanja podatkov.

2.1 Shema podatkov

Da bi naredili kakovosten načrt prilagoditve trenutne sheme podatkov na novo shemo podatkov, smo najprej naredili natančen pregled obstoječe sheme ETN podatkov. ETN podatki so podatki o poslih nepremičnin, ki so se sklenili. Vsak posel je sestavljen iz ene ali več nepremičnin. Podatki so v okviru obstoječe aplikacije shranjeni v petih različnih tabelah v podatkovni bazi; šifranti, posli, zemljišča, stavbe in deli stavb. Naredili smo primerjavo obstoječe sheme podatkov z novo shemo podatkov. Pri pregledu smo si pomagali s specifikacijo obeh shem, ki smo ju prejeli od GURS-a.

Na začetku smo opazili, da GURS v novi shemi ne vodi le kupoprodajnih

poslov, temveč tudi najemne posle. Ob nadaljnem pregledu smo ugotovili, da podatke o šifrantih za kupoprodajne in najemne posle lahko hranimo v eni tabeli, saj je shema podatkov o šifrantih enaka pri obeh vrstah poslov. Razlika je v nepremičninah, ki predstavljajo dele poslov. Obstoječa shema podatkov vsebuje tri vrste nepremičnin, in sicer stavbe, deli stavb in zemljišča. Ugotovili smo, da GURS podatkov za stavbe ne vodi več in podatke za stavbe beleži kar kot deli stavb. Obstoječa aplikacija podpira le kupoprodajne posle, ki so sestavljeni iz dveh vrst nepremičnin, in sicer delov stavb in zemljišč. Bistvena razlika, ki smo jo opazili je tudi ta, da je shema podatkov za dele stavb za kupoprodajne posle različna od najemnih. Vrsta nepremičnine zemljišča se nahaja le v kupoprodajnih poslih, najemni posli ne vsebujejo zemljišč. Ugotovili smo, da se podatki za najemne posle razlikujejo po shemi od kupoprodajnih poslov. Shema podatkov za kupoprodajne posle hrani nekaj dodatnih in tudi drugačnih podatkov kot o najemnih poslih. To pomeni, da kupoprodajne posle ni mogoče hraniti skupaj z najemnimi posli. Zaključili smo, da bo zaradi drugačne sheme podatkov in ukinitve nekaterih vrst podatkov, potrebno ustvariti nove tabele za hranjenje ETN podatkov v podatkovni bazi.

2.2 Pridobivanje podatkov

Aplikacija Cenilci je namenjena poizvedovanju in prikazovanju poslov nepremičnin. ETN podatke za prikaz v aplikaciji se pridobi iz GURS-a. Glede na to, da se vsak dan vnašajo novi posli, je tako ETN podatkov vedno več. Obstoječi sistem pridobivanja podatkov je tak, da GURS posreduje podatke nekajkrat na leto skrbniku aplikacije, ki jih s pomočjo programa za uvoz podatkov uvozi v podatkovno bazo. Iz tega dejstva je mogoče sklepati, da uporabniki v danem trenutku najverjetneje nimajo na voljo najnovejših podatkov o poslih s trga nepremičnin in s tem vpogleda v aktualne cene s trga nepremičnin. Podatki, ki jih GURS posreduje, so v tekstovni obliki. Vsaka vrsta podatkov je zapisana v svoji datoteki na CD mediju. Za uvoz podat-

kov v takšni obliki, je bila razvita posebna aplikacija za uvoz podatkov v podatkovno bazo. Aplikacija iz datotek prebere vrednosti podatkov o poslih in nepremičninah, jih pretvori v ustrezno shemo podatkov in jih shrani v podatkovno bazo.

2.3 Prikaz nepremičnin

Pri prikazovanju podatkov je pomemben tudi način prikaza, saj to doprinese k boljši uporabniški izkušnji, in sicer tudi v primeru prikaza nepremičnin. Obstoječa aplikacija podpira prikaz podatkov v obliki tabel. Primer prikaza rezultatov iskanja je na Sliki 2.1.

Id	Vrsta posla	Datum skl.	Pog. cena (EUR)	DDV	Net. tl. povr.	Upor. povr.	Prod. povr.	Prod. delež	Leto izgr.	Zazid. povr.	Kat. občina	Naslov
668442	ProTrg	07.11.2013	98.000,00									
668442	DS				78,30	78,30	78,25		1979	0,00	BREZOVICA	PODPEŠKA CESTA 152
668442	DS				60,00	60,00	58,27		1979	0,00	BREZOVICA	PODPEŠKA CESTA 152
674190	ProTrg	04.12.2013	52.000,00		48,60	66,20	66,00		1935	0,00	JEZERO	JEZERO 28
674883	ProTrg	23.12.2013	75.000,00		74,80	112,30	112,30		2009	0,00	BREZOVICA	NOVA POT 127 B
675532	ProTrg	08.01.2014	91.350,00		74,80	112,30	112,30		2009	0,00	BREZOVICA	NOVA POT 127 E
682092	ProTrg	12.03.2014	63.200,00									
682092	DS				44,30	54,90	19,00		1974	0,00		PLEŠIVICA 64
682092	DS				50,40	74,60	75,00		1974	0,00		PLEŠIVICA 64
684785	ProTrg	06.02.2014	93.364,00		100,00	100,00	115,00		1990	0,00		PODSVETIJA 28 A
692899	ProTrg	16.05.2014	80.000,00		110,10	63,50	63,00		1980	0,00	PRESERJE	PRESERJE 34
702869	ProTrg	22.07.2014	73.000,00		122,20	122,20	122,00		2004	0,00	PRESERJE	PODPEČ 48

Slika 2.1: Prikaz poslov in nepremičnin v seznamu v obstoječi aplikaciji.

V obstoječi aplikaciji je na voljo tudi podroben prikaz ETN podatkov za nepremičnine. Primer prikaza nepremičnine - del stavbe je prikazan na Sliki 2.2.

Trenutna aplikacija prikaže nepremičnine, ki sestavljajo najdene posle tudi na *Google Maps* zemljevidu. Primer prikaza se nahaja na Sliki 2.3. Tako ima uporabnik na voljo z zastavico (zelena pika) označeno lokacijo nepremičnine. Tak podatek omogoča lažjo oceno vrednosti nepremičnine, saj na višino cene običajno vpliva tudi lokacija nepremičnine. Npr. v bližini mest je neko zemljišče običajno vredno več, kot če je od mesta bolj oddaljeno. Vsaka zastavica na zemljevidu predstavlja eno nepremičnino. Ob kliku nanjo, ima uporabnik na voljo nekaj osnovnih informacij o poslu, h kateremu

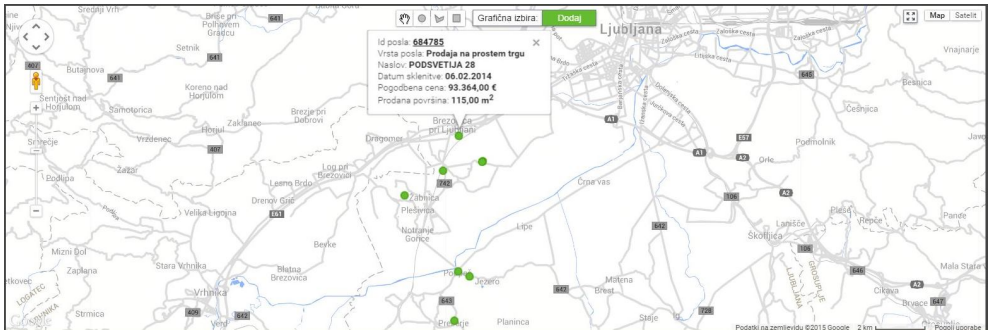
Podatki o delu stavbe

	ETN
Datum zadnje posodobitve:	
Vrednost nepremičnine:	
Število zapisov REN:	
Šifra katastrske občine:	1724
Katastrska občina:	BREZOVICA
Številka stavbe:	2559
Številka dela stavbe:	3
Občina:	BREZOVICA
Naselje:	VNANJE GORICE
Naslov dela stavbe:	PODPEŠKA CESTA
Hišna številka dela stavbe:	152
Številka stanovanja / poslovnega prostora:	
Katastrski vpis:	
Dejanska raba dela stavbe:	
Upravnik stavbe:	
Številka etaže:	
Dve ali več etaž:	
Lega v stavbi:	pritličje
Številka nadstropja:	
Atrij:	NE
Leto obnove oken:	
Leto obnove inštalacij:	
Obstoj klima naprav:	
Uporabna površina dela stavbe [m2]:	78,30
Neto tlorisna površina dela stavbe [m2]:	78,30
Posebna nepremičnina:	
Parkirišče:	
Stanovanje je v skupni lasti:	
Nestanovanjski del je v skupni lasti:	
Kuhinja:	
Kopalnica:	
Stranišče:	
Število sob:	0
Seznam drugih prostorov:	
Počitniška raba dela stavbe:	
Dejavnost prijavljena na naslovu:	
Vrsta najema:	
Vrsta dela stavbe:	Gostinski lokal
Prodani delež:	1/1

Slika 2.2: Prikaz podrobnosti dela stavbe v obstoječi aplikaciji v info listu.

pripada nepremičnina. Če je bila nepremičnina prodana večkrat, so izpisani tudi id-ji poslov, s katerimi je nepremičnina povezana.

Uporabnik ima na Google Maps zemljevidu možnost pogleda z Google Street View [15], ki omogoča dejanski pregled okolice. V desnem spodnjem kotu se nahaja majhen rumen možicelj, t.i. 'Pegman'. Uporabnik mora nanj najprej klikniti in ga nato povleči na zemljevid. Tiste ulice, ki so bile fotografirane, se obarvajo modro. Na tistem delu obarvane ulice ali ceste, kjer uporabnik spusti rumeno figuro oz. možiclja, bo vstopil v ulični pogled.



Slika 2.3: Prikaz delov poslov - nepremičnin na Google Maps zemljevidu.

Slika 2.4: Obstoječi prikaza nepremičnine na zemljevidu s pomočjo Google Street View storitve.

Možiclja je potrebno spustiti na modro obarvani ulici ali cesti. Da se prikaže pogled najbližje nepremičnini, je potrebno spustiti možiclja najbližje lokaciji nepremičnine. Ta del postopka je prikazan na Sliki 2.4. Ko uporabnik vstopi v ulični pogled, mora še usmeriti pogled iz privzete usmerjenosti v smer nepremičnine. Obstoječi postopek od uporabnika zahteva kar nekaj korakov in spretnosti, da dobi posnetek nepremičnin(e).

Poglavje 3

Opis problema

3.1 Konec pridobivanja podatkov po stari shemi podatkov

Obstoječa aplikacija operira z ETN podatki, ki so shranjeni v podatkovni bazi. Podatki so bili v podatkovno bazo uvoženi v začetku leta. To pomeni, da je preteklo že kar nekaj časa od zadnje posodobitve podatkov, zato uporabniki nimajo na voljo aktualnih/svežih podatkov s trga nepremičnin. Uvoz podatkov s programom za uvoz iz datotek je bil tudi zadnji po stari shemi podatkov, saj GURS omogoča posredovanje podatkov le še po novi shemi podatkov. Analiza obstoječe in nove sheme podatkov je pokazala kar nekaj razlik med shemama. Ugotovili smo, da GURS podatkov za stavbe ne vodi več in podatke za stavbe beleži kar kot dele stavb. To pomeni, da bo potrebna ukinitev prikaza vrste nepremičnin stavbe. Na podlagi ugotovitev razlik med shemama podatkov smo spoznali, da je prilagoditev aplikacije na novo shemo podatkov potrebna čimprej. Če aplikacije ne prilagodimo na novo shemo podatkov, uporabniki ne bodo imeli več na voljo novih ETN podatkov. Tako bo aplikacija postala vse manj uporabna.

3.2 Obstoječa aplikacija ne vsebuje najemnih poslov

V obstoječi aplikaciji imajo uporabniki na voljo le kupoprodajne posle. Pri pregledu razlik med staro in novo shemo podatkov smo opazili, da je v novi shemi podatkov na voljo tudi nova vrsta poslov - najemni posli. Ta vrsta poslov je sestavljena le iz delov stavb. Shemi podatkov o delih stavb za najemne in kupoprodajne posle pa se med seboj razlikujeta. To pomeni, da bo potrebno posebej hraniti podatke o najemih za dele stavb in podatke o poslih. Prav tako tudi podatke o poslih; za kupoprodajne posle posebej, za najemne posle posebej, saj se shemi podatkov o omenjenih vrstah poslov razlikujeta. V preteklosti GURS podatkov o najemnih poslih ni zbiral in je s tem pričel šele pred kratkim. Glede na to, da so uporabniki aplikacije cenilci nepremičnin, bo podpora nove vrste poslov izboljšala uporabniško izkušnjo za trenutne uporabnike ter še povečala uporabnost aplikacije. Zaradi večjega nabora in raznolikosti podatkov, bo aplikacija kot pripomoček za svoje delo uporabna tudi za druge, nove uporabnike. Aplikacija bo tako postala uporabna tudi za takšne vrste uporabnikov, ki se ukvarjajo npr. s posredništvom nepremičnin, ki delajo cenitve nepremičnin za npr. poslovne prostore, garaže in ne več le za tiste, ki se ukvarjajo s cenitvami nepremičnin, namenjenih prodaji.

3.3 Neuporabnost obstoječega programa za uvoz podatkov iz nove sheme podatkov

Obstoječi način pridobivanja podatkov smo že pogledali, zato je ponovno opisovanje postopka odveč. Iz opisa razlik med novo in staro shemo podatkov ter opisa obstoječega načina pridobivanja podatkov v prejšnjem poglavju je razvidno, da obstoječi program za uvoz ETN podatkov v podatkovno bazo ne podpira uvoza podatkov iz nove sheme podatkov. Trenutni program za uvoz podatkov podpira le uvoz podatkov po stari shemi podatkov, nova pa

se precej razlikuje od stare.

Glavne razlike med shemama podatkov so naslednje:

- vrsta nepremičnin - stavbe ni več na voljo,
- nova vrsta poslov - najemni posli; shemi podatkov za kupoprodajne posle in najemne posle se razlikujeta,
- shemi podatkov za kupoprodajne dele stavb in najemne dele stavb se razlikujeta.

Iz razlik med shemama je razvidno, da je potrebno spremeniti in prilagoditi tudi obstoječi program za uvoz ETN podatkov v podatkovno bazo. Nov program mora podpirati uvoz podatkov o šifrantih, kupoprodajnih in najemnih poslih (vsakega posebej), nepremičninah - kupoprodajni deli stavb in zemljišča ter najemni deli stavb. Program za uvoz je odvisen od podatkovne baze, zato je bilo potrebno najprej pripraviti podatkovno bazo - tabele, kamor smo shranili ETN podatke. Obstoječi program za uvoz podatkov podpira tudi uvoz podatkov za stavbe. Te podpore v novem programu ne potrebujemo več, ker nova shema ne vsebuje podatkov za stavbe.

3.4 Zahteven način prikaza nepremičnine

Obstoječi način prikaza nepremičnin smo si pogledali v Poglavju 2. Ugotovili smo, da mora uporabnik skozi več korakov, ki mu omogočijo prikaz nepremičnine z Google Street View. Tak način prikaza nepremičnine je za uporabnika precej zahteven za uporabo. Še posebej v primerih, ko uporabnik za oceno neke nepremičnine najprej pregleda več deset nepremičnin, z namenom da si ustvari predstavo stanja nepremičnin na trgu v danem časovnem obdobju. Za to pa porabi kar nekaj časa. Opisan postopek/problem je dovolj zgovoren motiv za razvoj funkcionalnosti, ki avtomatizira in skrajša število korakov za prikaz nepremičnine. S takšno implementacijo nove funkcionalnosti, precej pripomoremo k izboljšanju uporabniške izkušnje aplikacije. Ta

funkcionalnost bo uporabnikom omogočila hitrejši pregled nepremičnin. Zato smo se odločili, da bomo izdelali prototip funkcionalnosti prikaza nepremičnin z uprorabo uličnega pogleda z enim klikom na nepremičnino. Prototip funcionalnosti bomo razvili s pomočjo Google Street View API-ja.

3.5 Konec obstoječega načina pridobivanja podatkov z GURS-a

Aplikacija je namenjena prikazovanju podatkov o poslih nepremičnin. Najbolj željeni podatki za prikaz so podatki o poslih, ki so se sklenili pred kratkim. Iz opisa obstoječega stanja je mogoče sklepati, da način pridobitve ETN podatkov ne omogoča uporabnikom v vsakem trenutku pregled podatkov o svežih poslih. Uporabniki tako nimajo na voljo podatkov o aktualnem stanju cen na trgu nepremičnin. Zaradi obstoječega stanja uporabniki običajno operirajo s kar nekaj meseci starimi podatki. To situacijo bi bilo mogoče izboljšati z dodatnimi dogovori, da GURS vsak mesec posreduje podatke o poslih, ki so bili vnešeni v ETN pretekli mesec. Tako bi uporabniki imeli podatke, stare največ en mesec. Po povpraševanju na GURS za pogostejše posredovanje podatkov smo dobili odgovor, da ukinjajo obstoječi način posredovanja podatkov. To pomeni, da posredovanje podatkov v obliki datotek v bližnji prihodnosti ne bo več mogoče. Rešitev, ki so jo ponudili je, dodeljen dostop do GURS spletnega servisa, preko katerega je mogoče pridobivati ETN podatke. Rešitev GURS-a, ki omogoča prenos podatkov v vsakem trenutku, je bil dovolj dober razlog za implementacijo prototipa funkcionalnosti. Ta bo omogočal avtomatsko prenašanje podatkov o poslih in shranitev v podatkovno bazo aplikacije. Takšna rešitev omogoča uporabnikom, da imajo na voljo v vsakem trenutku sveže ETN podatke.

Poglavje 4

Uporabljena programska orodja in tehnologije

4.1 Microsoft SQL Server Management Studio 2012

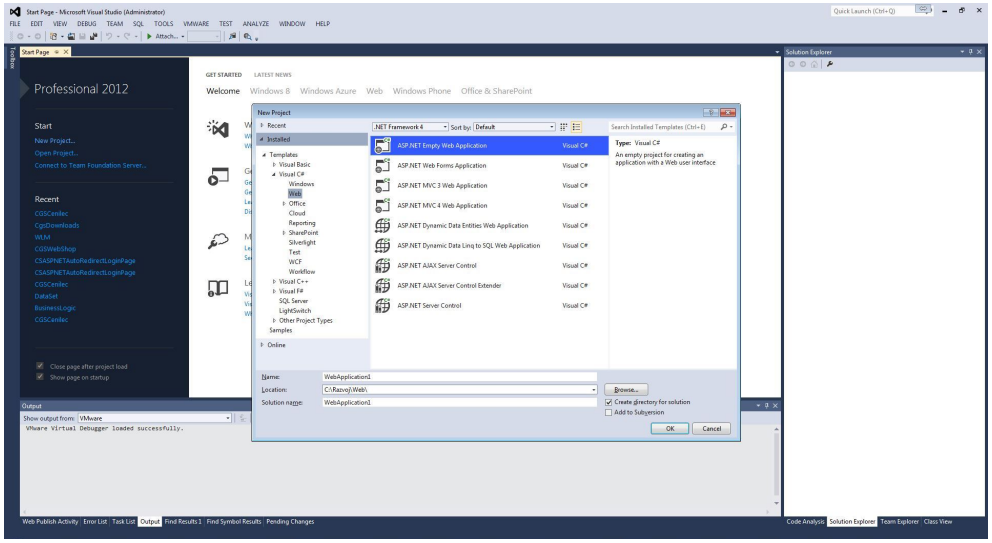
SQL Server Management Studio je grafični uporabniški vmesnik za upravljanje in administracijo Microsoft SQL Server podatkovnih baz. Program smo uporabljali za ustvarjanje kopij podatkovne baze, ustvarjanje novih in spreminjanje obstoječih tabel, ter baznih procedur. Uporabljali smo ga tudi za pregledovanje podatkov v podatkovni bazi.

4.2 Microsoft Visual Studio 2012

Visual Studio je razvojno okolje, ki ga je razvilo podjetje Microsoft. Namenjeno je razvoju spletnih in namiznih aplikacij. Orodje je zelo preprosto za uporabo. Pri razvoju aplikacije je kontolnike mogoče hitro nanesti na delovni površino, nakar z dvojnim klikom nanje že lahko dostopamo do avtomatsko ustvarjenih funkcij, kot je naprimer klik na nek gumb. Omogoča tudi samodejno dokončanje ukazov z razširitvijo *InteliSense*. Visual Studio

omogoča razvijalcu urejanje kode, urejanje grafičnega uporabniškega vmesnika ter podpira razhroščevalnik. Vsebuje tudi prevajalnik programske kode.

Visual Studio smo uporabljali za urejanje programske kode pri prilagajanju programa za uvoz podatkov, prilagajanje aplikacije na novo shemo podatkov, razvoj prikaza nepremičnin in pridobivanje podatkov prek spletnega servisa.



Slika 4.1: Microsoft Visual Studio 2012.

4.3 ASP.NET

ASP.NET je tehnologija, ki se izvaja na strani strežnika. Tehnologijo ASP.NET je razvilo podjetje Microsoft za namen razvijanja spletnih aplikacij. Tehnologija je izdana v paketu z .NET Framework-om [7]. Je odprtokodno, strežniško orientirano programsko ogrodje, ki se uporablja za izdelavo dinamičnih spletnih strani. Ogrodje ASP.NET je zgrajeno z uporabo prevajalnika CLR, kar omogoča programerjem pisanje v katerem koli podprtem .NET programskem jeziku. .NET programski jeziki so [5]: Visual C++, Visual C#, Visual Basic, Visual F#... Zadnja izdana verzija .NET Frameworka je 4.6 RTM.

4.4 C#

C# (C sharp) [8] je eden od programskih jezikov ogrodja ASP.NET. Je moderen jezik, na visoki ravni. Za razvoj se uporablja razvojno okolje Visual Studio in .NET Framework [5]. Je objektno usmerjen [1], preprost, močan in varen jezik. Novosti v jeziku C# omogočajo razvijalcem hiter razvoj aplikacij, hkrati pa ohranjajo eleganco in izraznost stilov jezika C.

4.5 HTML

HTML (ang. Hyper Text Markup Language) [9] je spletni jezik, ki predstavlja osnovo vseh spletnih strani. Je najosnovnejši in najbolj razširjen označevalni jezik za izdelavo spletnih strani. Z jezikom HTML [6] v brskalniku določimo postavitev elementov na strani. Jezik HTML uporablja elemente, ki so sestavljeni iz značk. Značke so običajno zapisane v parih, ki označujejo začetek in konec določenega dokumenta. Znotraj značk je mogoče tudi gnezdenje drugih značk. Z nastavitvami lastnosti značk je mogoče nastaviti stil izpisa, kot je npr. barva besedila, barva ozadnja, vrsta pisave, velikost izpisa, poravnava besedila... HTML podpira tudi značke, ki določajo obliko prikazane vsebine znotraj značk. Besedilo je lahko prikazano kot odebeljeno, ležeče, podčrtano, pomembno, označeno, poudarjeno... S HTML značkami je mogoče besedilo (in tudi ostalo vsebino) prikazati kot našteto po alinejah, zapisano v tabeli, URL povezava, citirano besedilo... HTML omogoča tudi prikaz slik. Najnovejša različica HTML5 omogoča poenostavljen način zapisa značk (npr. način kodiranja znakov), podprte so nove oz. dodatne značke. Posebej pomembna je podpora multimediji (predvajanje avdio in video), ki vse bolj nadomešča predvajalnik *Flash* ter možnost risanja grafik s pomočjo jezika *JavaScript*.

4.6 JavaScript

JavaScript [11] je programski jezik, ki se ga uporablja v spletnih aplikacijah in se izvaja na strani uporabnika v spletnem brskalniku. Je objektno skrip-

tni jezik, ki je bil razvit z namenom, da pomaga spletnim programerjem pri ustvarjanju interaktivnih spletnih strani. JavaScript si s programskim jezikom Java deli številne lastnosti in shemo. JavaScript sintaksa je podobna sintaksi jezika C, le da nima vgrajenih vhodno izhodnih funkcij. Veliko se uporablja pri izdelavi dinamičnosti spletnih strani. Jezik je podprt v skoraj vseh spletnih brskalnikih. Program se vgradi ali vključi v HTML.
Z JavaScript jezikom smo v aplikaciji izvedli, med drugim tudi pravilno delovanje iskalne plošče. Na ta način smo onemogočili izbiro za iskanje po najemnih zemljiščih.

4.7 SQL

Jezik SQL (angl. Structured Query Language) [13] ali strukturirani povpraševalni jezik je namenjen poizvedovanju in pridobivanju podatkov iz podatkovne baze. Uporablja se za upravljanje podatkov v relacijskih podatkovnih bazah in za obdelavo toka podatkov v sistemu za upravljanje relacijskih tokov podatkov [12]. Jezik prvotno temelji na relacijski algebri. Sestavljen je iz jezika za definiranje podatkov, manipulacijo s podatki in nadzor podatkov. SQL jezik [4] vsebuje operacije kot so: izberi, vstavi, ustvari, izbriši, posodobi, popravi. V splošnem je deklarativni jezik, vendar vključuje tudi postopkovne elemente.

Poglavje 5

Razvoj novih funkcionalnosti aplikacije in integracija nove sheme podatkov

V tem poglavju bomo spoznali postopek prilagoditve obstoječe sheme na novo shemo podatkov. Predstavili bomo tudi razvoj prototipov novih funkcionalnosti, in sicer prikaz nepremičnin s pomočjo *Google Street View API*-ja in avtomatski prenos podatkov prek spletnega servisa.

5.1 Prilagoditev obstoječega sistema na novo shemo podatkov

5.1.1 Prilagoditev podatkovne baze - priprava tabel

Z analizo obstoječega sistema in primerjavo trenutne sheme podatkov z novo smo ugotovili, da bo potrebno spremeniti podatkovno bazo, kjer hranimo podatke ETN. Zaradi prilagajanja smo najprej naredili testno podatkovno bazo kot kopijo obstoječe, z namenom da se izognemo nedelovanju obstoječe aplikacije. Vse nadaljne spremembe smo izvajali na testni podatkovi bazi. Zaradi ukinitve vrste nepremičnine stavbe smo tabelo, kjer hranimo podatke

o stavbah, izbrisali. Ker smo v sistemu podprli novo vrsto poslov - najemni posli, smo iz ugotovitve, da ni mogoče shraniti v isto tabelo podatkov o obeh vrstah poslov sklenili, da ustvarimo za vsako vrsto poslov svojo tabelo. Podobno smo naredili tudi za dele stavb, saj se podatki o delih stavb v kupoprodajnih poslih razlikujejo od najemnih delov stavb. Zemljišča so podprta le v kupoprodajnih poslih. Iz analize o razlikah med shemama smo povzeli, da bomo za shranitev ETN podatkov potrebovali naslednje tabele:

- šifranti - *sifranti*,
- kupoprodajni posli - *posliKPP*,
- najemni posli - *posliNP*,
- kupoprodajni deli stavb - *deliStavbKPP*,
- najemni deli stavb - *deliStavNP*,
- kupoprodajna zemljišča - *zemljiscaKPP*.

V testni podatkovni bazi smo ustvarili zgoraj naštete tabele in jih ustrezno poimenovali. Vsaki izmed tabel smo dodali še opis tabele. Tabelam smo nastavili atribute in njihove podatkovne tipe. Pri nastavljanju atributov in njihovih podatkovnih tipov smo se zgledovali po specifikaciji nove sheme podatkov. Za pregledovanje sheme podatkovne baze smo uporabljali program *SQL Server Management Studio*. Z omenjenim programom smo ustvarili testno podatkovno bazo, urejali tabele (dodajali nove, odstranjevali stare, urejali atribute tabel) in urejali bazne procedure (spreminjali, dodajali in brisali). Tabelam smo nastavili tudi primarne ključe (ang. Primary Keys) in relacije med tabelami.

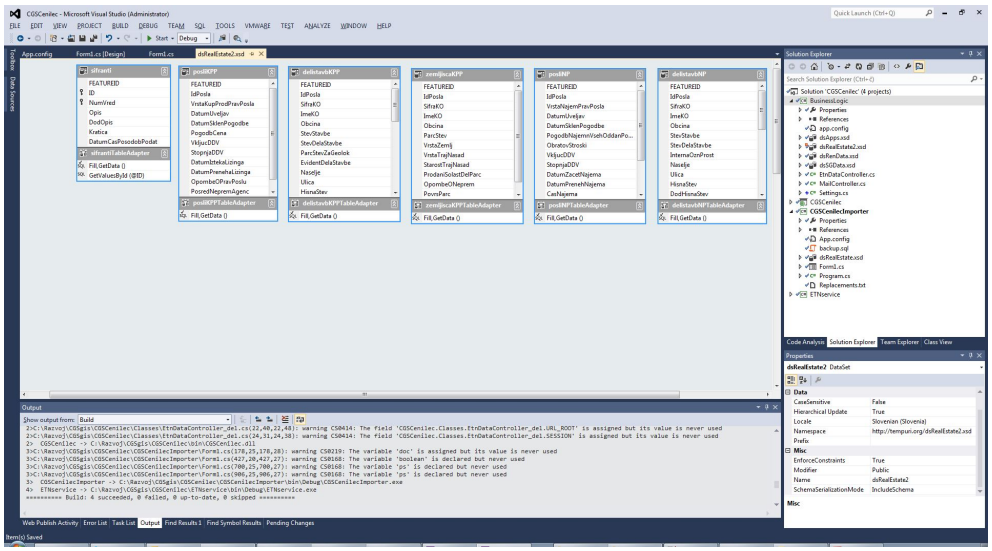
5.1.2 Prilagoditev programa za uvoz podatkov na novo shemo podatkov

Aplikacija upravlja s podatki ETN, ki jih pridobimo z GURS-a, kar smo že pogledali v prejšnjih poglavjih. Za potrebe sprotnega testiranja pravilnega

delovanja aplikacije pri prilagajanju aplikacije na novo shemo podatkov, smo potrebovali tudi testne podatke. Ker sistem avtomatskega pridobivanja podatkov preko spletnega servisa še ni bil vzpostavljen, smo pridobili še zadnji paket podatkov po starem načinu od GURS-a. To smo storili po že opisanem načinu - uvoz podatkov ETN iz datotek s programom za uvoz podatkov v podatkovno bazo aplikacije. V prejšnjih poglavjih smo ugotovili, da za uvoz podatkov po obstoječem načinu, torej s programom za uvoz podatkov iz datotek, potrebujemo program za uvoz podatkov, ki podpira novo shemo podatkov. Vse to so razlogi, da smo najprej naredili prilagoditev programa za uvoz podatkov iz datotek na novo shemo podatkov. Podatkovno bazo - tabele smo že pripravili, tako da smo se v tem koraku lahko posvetili spreminjanju programske kode programa za uvoz podatkov. Za urejanje programske kode obstoječega programa za uvoz podatkov iz datotek v podatkovno bazo smo uporabljali program *Visual Studio*.

Za povezavo programa do podatkovne baze, smo najprej nastavili povezavni niz (ang. Connection String), in sicer v projektu aplikacije. Nastavili smo ime podatkovne baze, način avtentikacije, uporabniško ime in geslo za dostop programa do podatkovne baze. V obstoječi projekt DataSet, ki je namenjen delu s podatkovno bazo, smo dodali tabele, ki smo jih ustvarili v podatkovni bazi. Tabele iz podatkovne baze v Visual Studiu se imenujejo 'TableAdapterji'. Primer TableAdapterjev na Sliki 5.1. S tem, ko smo dodali tabele v projekt, so se avtomatsko prenesli tudi atributi in njihovi podatkovni tipi.

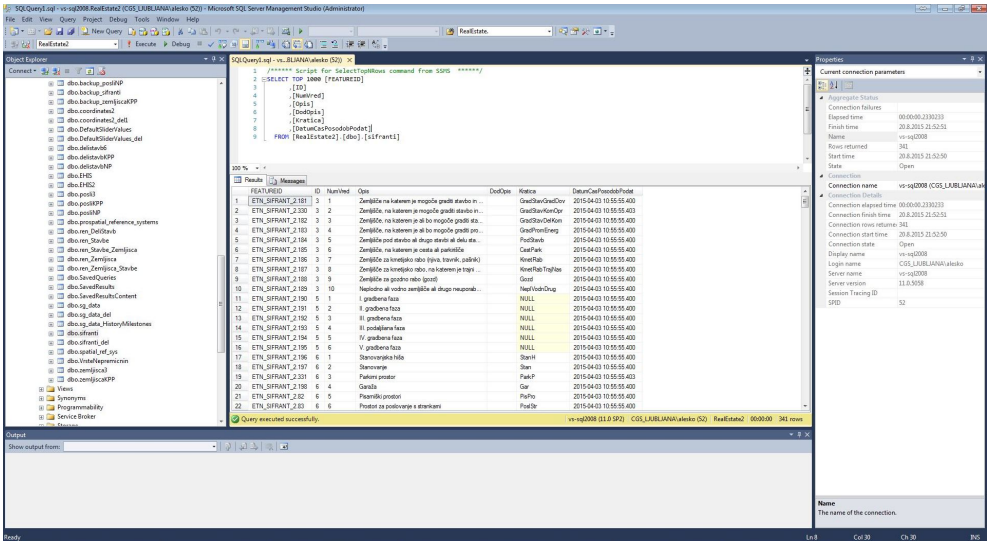
V naslednjem koraku prilagajanja programa za uvoz smo pričeli s prilagajanjem programskega dela aplikacije. Zaradi ukinitve vrste nepremičnine stavbe smo obstoječo funkcijo za uvoz stavb odstranili, saj je ne potrebujemo več. Po pregledu razlik med shemama smo se odločili, da za vsako vrsto podatkov (tabelo) napišemo funkcijo, s katero bomo uvozili podatke iz datoteke v posamezno tabelo. Napisali smo šest različnih funkcij; eno za šifrante, dve za posle in tri za nepremičnine. Logiko branja podatkov iz datotek, obdelavo podatkov in pisanja podatkov v podatkovno bazo smo ohranili



Slika 5.1: Tabele v Data Setu v programu Microsoft Visual Studio.

iz starih funkcij. Za vsako vrsto podatka (t.j. posli, nepremičnine, šifranti), obstaja posamezna datoteka z ETN podatki. Datoteke so sestavljene iz več vrstic. Vzemimo, da ena vrstica predstavlja en objekt. Objekt je sestavljen iz več atributov, vsak atribut pa ima svojo vrednost. En objekt s podatki predstavlja podrobnosti enega posla oz. nepremičnine oz. šifranta. Atributi so v vrsticah ločeni z znakom ';'. Prva vrstica v datoteki predstavlja nazive atributov, vse ostale vrstice pa predstavljajo vrednosti atributov. Iz prve vrstice v datoteki smo razbrali število atributov, ki predstavljajo en objekt. Prebrana vrstica je tipa besedilo (ang. String), prav tako tudi razdeljene vrednosti po znaku ';'. Ker ima vsak atribut določen podatkovni tip, smo vrednosti atributov sproti pretvarjali v ustrezen podatkovni tip. Podatkovni tipi atributov so določeni po specifikaciji nove sheme podatkov. Prebrano datoteko smo, razdeljeno po vrsticah, obdelali v zanki. V zanki smo ustvarjali objekte in nastavljali vrednosti atributom objekta, pri čemer smo objekte sproti shranjevali v posebno spremenljivko. Po končanem branju in nastavljanju, smo prebrane podatke z ukazom *Update* zapisali v podatkovno bazo v ustrezno tabelo. Ko smo imeli pripravljen program za uvoz vseh vrst podat-

kov, smo naredili še uvoz podatkov v podatkovno bazo. Pri prvem poskusu uvoza podatkov smo naleteli na kar nekaj težav. Odkrili smo, da je do napak v programu prišlo zaradi nepravilne in pomanjkljive specifikaciji sheme podatkov, ki smo jo prejeli z GURS-a. Pri nastavljanju vrednosti atributom objektov v tabelah smo se zgledovali po specifikaciji. Izkazalo se je, da podatki niso pričakovanih podatkovnih tipov. Ko smo napake odpravili, smo ponovno poskusili z uvozom in tako podatke uspešno uvozili v podatkovno bazo. Prikaz uvoženih podatkov o šifrantih na Sliki 5.2.

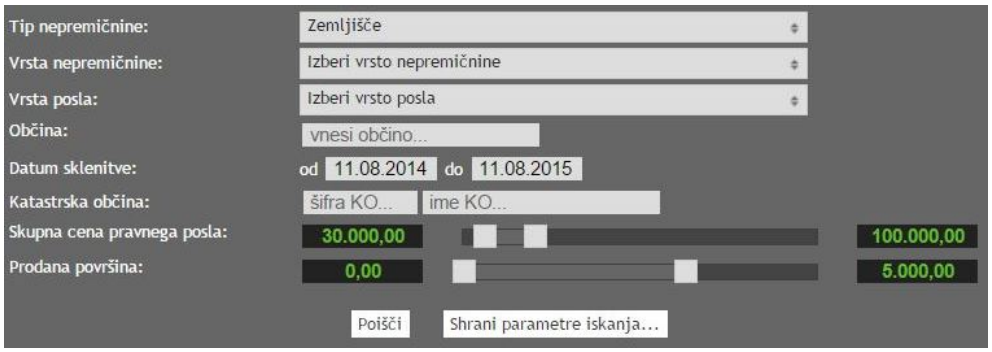


Slika 5.2: Uvoženi podatki v podatkovni bazi.

5.1.3 Prilagoditev programske logike aplikacije

Po uvozu podatkov v podatkovno bazo smo pričeli s prilagajanjem programske kode aplikacije na novo shemo podatkov. V programski kodi smo poiskali vse funkcije, ki manipulirajo s podatki o šifrantih in jih ustrezno popravili. Po ugotovitvi razlike med shemama, da v novi ni več podatkov o stavbah, smo pričeli z ukinitvijo vodenja podatkov o stavbah tudi v programski kodi. Stavbe sicer GURS še vedno vodi kot nepremičnino v poslu, vendar jo hrani kot del stavbe. Začeli smo s prilagajanjem iskalnega obrazca na Sliki 5.3.

Z iskalne plošče smo najprej umaknili podporo iskanja poslov nepremičnin - vrsta stavba. Zaradi dejstva, da sedaj obstajajo poleg podatkov za kupoprodajne posle tudi podatki za najemne posle, smo v iskalni obrazec dodali še to možnost.



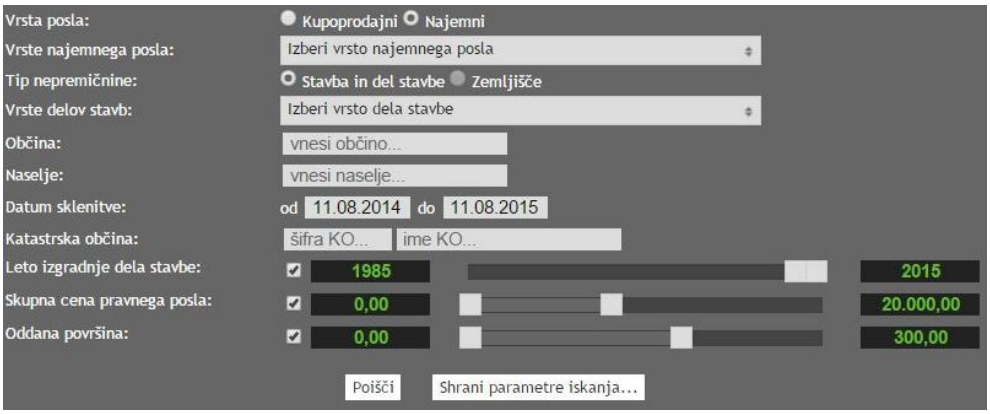
Slika 5.3: Iskalni obrazec v obstoječi aplikaciji.

V iskalnem obrazcu smo podprli naslednje načine iskanj:

- Iskanje kupoprodajnih poslov:
  - Iskanje delov stavb,
  - Iskanje zemljišč.
- Iskanje najemnih poslov:
  - Iskanje delov stavb.

Glede na to, da zemljišč najemni posli ne vsebujejo, smo iskanje po najemnih zemljiščih onemogočili. V zaledni programski kodi smo sicer obravnavali tudi omenjeno iskanje. Za take primere uporabniku izpišemo le obvestilo, da takšno iskanje ni mogoče. V iskalni obrazec na Sliki 5.4 smo dodali še nekaj dodatnih možnosti pri iskanju. Slednje uporabniku omogočajo natančnejše definiranje iskanih poslov.

Po končanem prilagajanju iskalnega obrazca smo pričeli s spremembo glavne fukcionalnosti aplikacije, in sicer s prilagajanjem funkcije za iskanje



Slika 5.4: Prilagojen iskalni obrazec.

poslov. V prvem koraku smo odstranili iskanje po stavbah. Obstoječa aplikacija deluje tako, da za vsako vrsto nepremičnine obstaja razred. Uporablja se ga pri nastavljanju vrednosti za posamezen element, t.j. posel ali nepremičnina. Zaradi ukinitve vodenja nepremičnine stavbe kot del posla, smo najprej odstranili razred *stavba*. Program Visual Studio nam je izpisal na katerih mestih so zaradi izbrisa omenjenega razreda nastale napake. V programu smo vse dele kode, ki se nanašajo na stavbe, odstranili ter tako ukinili podporo za nepremičnino stavba v aplikaciji. Nekatere funkcije smo izbrisali v celoti, druge pa smo le prilagodili. Zaradi integracije podpore iskanju po najemnih poslih smo dobro preučili obstoječi način pridobivanja kupoprodajnih poslov. Ugotovili smo, da podatkov o kupoprodajnih in najemnih poslov ne moremo pridobivati z enakimi funkcijami, prav tako pa ne ne moremo hraniti podatkov v istem razredu. Podobno velja tudi za dele stavb, ki predstavljajo dele posla. Podatki o delih stavb pri kupoprodajnih poslih so različni in jih ni mogoče shranjevati v isto shemo oz. razred, ker bi sicer razred moral vsebovati veliko različnih atributov. Nepremičnine zemljišča so prisotna le v kupoprodajnih poslih, zato smo za zemljišča uporabili en razred.

Ustvarili smo naslednje razrede, ki smo jih uporabljali za začasno hranjenje podatkov:

- Razred za kupoprodajne posle - *poselKPP*,
- Razred za najemne posle - *poselNP*,
- Razred za kupoprodajne dele stavb - *deliStavbKPP*,
- Razred za najemne dele stavb - *deliStavbNP*,
- Razred za kupoprodajna zemljišča - *zemljiscaKPP*.

Za iskanje smo ustvarili tri različne funkcije, za vsak način iskanja po eno. Obstoječe funkcije za iskanje smo uporabili le za zgled. Za iskanje in pridobivanje podatkov iz baze smo uporabljali bazne procedure (ang. Stored Procedures). Več o store procedurah sledi v nadaljevanju. Po pridobitvi podatkov iz podatkovne baze je potrebno podatke še obdelati, tj. posle z enakimi id-ji 'združiti' v enega, zadetke ustrezno urediti itd... Za vsak posel smo ustvarili novo instanco razreda (poselKPP oz. poselNP) in vsaki instanci posla nastavili vrednosti atributom. V spremenljivko (seznam) 'deli posla' smo shranili vse nepremičnine, ki pripadajo poslu. Za vsako nepremičnino posla smo (podobno kot za posel) ustvarili novo instanco ustreznega razreda (delStavbeKPP, zemljisceKPP, delStavbeNP), nastavili vrednosti atributom in shranili v seznam nepremičnin/delov posla. Del obdelave podatkov smo programirali v okolju *ASP.NET* v programskem jeziku *C#*. Omenjeno tehnologijo in programski jezik sem omenil v Poglavju 4. Sledil je še korak prilagoditve prikaza podatkov uporabniku. Tudi v tem koraku prilagoditve aplikacije smo se zgledovali po obstoječem načinu prikazovanja podatkov. Funkcije smo morali zaradi drugačnih podatkov o poslih in nepremičninah prilagoditi. Podatki, ki so bili pridobljeni in pripravljeni s strani uporabnika, se prikažejo na zaslonu. Zaradi ukinitve vrste nepremičnin stavbe smo nekatere dele kode tudi tu le izbrisali, nekatere pa prilagodili. Večino funkcij je bilo treba podvojiti, zaradi različnih shem podatkov o kupoprodajnih

in najemnih poslih. Za prikaz podatkov uporabnikov smo uporabili jezik *HTML*, za obdelavo podatkov na uporabnkovi strani pa programski jezik *JavaScript*. Ker podatki o poslih in nepremičninah v novi shemi podatkov vsebujejo nekatere podatke, ki jih obtoječa shema ni vsebovala, smo dodali polja za prikaz podatkov. Nekatera polja pa smo zaradi ukinitve nekaterih podatkov odstranili.

Aplikacija omogoča poleg iskanja tudi sortiranje rezultatov iskanja, izvoz rezultatov iskanja v excel dokument, podroben prikaz podatkov in nepremičnin posla - info list, iskanje poslov po id-ju posla, shranjevanje parametrov iskanja iz iskalne plošče in pridobivanje ter nastavljanje vrednosti nazaj, shranjevanje rezultatov iskanja in pridobivanje ter nastavljanje vrednosti nazaj. Vse funkcije smo prilagodili na novo shemo podatkov. Zaradi ukinitve nepremičnine stavbe in dodatne vrste poslov - najemi, smo naredili precej sprememb v funkcionalnosti. Najemni posli so vzrok, da se programska logika na nekaterih mestih deloma podvaja, saj je shema podatkov za kupoprodajne posle drugačna od sheme za najemne posle.

Po končanem prilagajanju programske kode aplikacije, smo želeli spremenjene funkcionalnosti stestirati, vendar smo hitro ugotovili, da prilagoditve aplikacije še ni konec. Preostalo nam je še prilagajanje baznih procedur, ki jih aplikacija uporablja za pridobivanje podatkov iz in vpisovanje v podatkovno bazo.

5.1.4 Prilagoditev podatkovne baze - spreminjanje in dodajanje novih baznih procedur

Za pridobivanje podatkov iz podatkovne baze se v aplikaciji uporablja bazne procedure (ang. Stored Procedures) [14]. Procedure spominjajo na funkcije v drugih programskih jezikih, saj lahko sprejmejo vhodne parametre in vrnejo več vrednosti v obliki izhodnih parametrov aplikaciji, ki je proceduro klicala. Bazne procedure vsebujejo programske stavke, ki izvajajo operacije v podatkovni bazi. Ti lahko kličejo tudi druge procedure in funkcije v podatkovni bazi. Procedura vedno vrne stanje programu, ki je proceduro poklical.

Postopek se je torej izvedel uspešno ali pa neuspešno. V tem primeru, procedura poda razlog za neuspeh. Bazne procedure smo v aplikaciji uporabljali za preproste in zahtevnejše poizvedbe. Primer preproste bazne procedure v Primeru kode 5.1.

```
1 CREATE PROCEDURE [dbo].[GetSetting]
2 (
3     @SettingId varchar(50)
4 )
5 AS
6     SET NOCOUNT ON;
7 SELECT *
8 FROM app_Settings
9 WHERE Id = @SettingId
```

Listing 5.1: Primer bazne procedure, ki vrne podatke o nastavitvah glede na podan parameter.

Klic baznih procedur v aplikaciji je precej preprost. Primer klica procedure v Primeru kode 5.2. V programski kodi smo prek *Table Adapter-ja* izvedli klic procedure s parametri, vrnjen rezultat pa nato obdelali s programsko logiko.

Zaradi ukinitve vrste nepremičnine stavba je bilo potrebno prilagoditi tudi bazne procedure. Izbrisali smo procedure, ki operirajo le nad podatki o stavbah. Tiste, ki operirajo tudi z drugimi podatki, kot so posli in nepremičnine, smo le popravili. Na vseh teh mestih smo odstranili sklicevanje na tabelo stavbe, saj smo jo predhodno že odstranili iz podatkovne baze. Zaradi podpore nove vrste poslov - najemni posli, je bilo potrebno dodati bazne procedure za pridobivanje podatkov o najemnih poslih in o delih stavb za najeme.

```
1  public static List<KeyValuePair> GetValuesOfSifrant(int id)
2  {
3      List<KeyValuePair> l = new List<KeyValuePair>();
4      var t = new sifrantiTableAdapter().GetValuesById(id);
5
6      foreach (var r in t)
7          l.Add(new KeyValuePair(r.NumVred, r.Opis, ↩
               r.IsKraticaNull() ? "" : ↩
               string.IsNullOrEmpty(r.Kratica) || ↩
               string.IsNullOrWhiteSpace(r.Kratica) ? "" : ↩
               r.Kratica));
8
9      return l;
10 }
```

Listing 5.2: Primer klica bazne procedure v funkciji, ki vrne podatke o nastavitvi glede na podan id.

Bazne procedure smo uporabljali za iskanje poslov in nepremičnin ter pridobivanje podatkov o le-teh. Poleg tega smo na ta način urejali rezultate iskanja, shranjevanje in pridobivanje (nazaj) parametrov iskanja, za shranjevanje in pridobivanja (nazaj) rezultatov iskanja, iskanje poslov po id-ju. Po koncu prilagajanja baznih procedur smo v programski kodi na mestih, kjer pridobivamo podatke iz podatkovne baze, spremenili/dodali klice ustreznih baznih procedur.

Nato smo pričeli s testiranjem aplikacije. Tudi tokrat se je pojavilo kar nekaj napak, t.i. *error-jev*. Največ napak je bilo na mestih, kjer smo nastavljali vrednosti instancam (primer razreda) razredov in instance niso imele vrednosti oz. so imele vrednost *null*. Do teh težav je prišlo, ker predhodno nismo naredili inicializacije spremenljivk. Napake so se pojavile tudi na mestih, kjer smo prirejali spremenljivkam vrednosti z napačnim podatkovnim tipom. Po odpravi napak smo še kar nekaj časa namenili testiranju. Tako smo se lahko prepričali v pravilnost delovanja aplikacije nad novo shemo podatkov.

5.2 Prototip prikaza nepremičnin z 'Google Street View'

Prikaz nepremičnine v obstoječi aplikaciji smo že spoznali v Poglavju 2. Ugotovili smo, da je glavni problem obstoječe rešitve dolg postopek do prikaza nepremičnine z rešitvijo Google Street View. Zato smo se odločili, da razvijemo prototip funkcionalnosti, ki poenostavljeno prikaže nepremičnino z Google Street View API-jem. Zaradi nepoznavanja API-ja ni bilo mogoče z gotovostjo trditi v kakšni meri bomo uspeli razviti rešitev. Definirali smo zahteve problema in jim skozi razvoj tudi sledili. Začetna ideja je bila, da izdelamo prototip funkcionalnosti, tako da z enim klikom v novem oknu prikaže panoramska slika nepremičnine z rešitvijo Google Street View. To je za nas predstavljajo najboljšo možno rešitev. To je bila pravzaprav raziskovanlna naloga, saj nismo poznali podobne, že implementirane rešitve. Poleg tega nismo poznali API-ja, zato ni bilo mogoče napovedat v kakšni meri bomo rešitev sploh lahko razvili. Na podlagi podatkov o lokaciji nepremičnine, zemljepisne širine (ang. latitude) in zemljepisne dolžine (ang. longitude), smo želeli prikazati nepremičnino v panoramskem pogledu uličnega pogleda.

5.2.1 Google Street View API

Google Street View je projekt podjetja Google za prikaz uličnega pogleda in izhaja iz projekta Google Maps zemljevidi. Google Street View [17] omogoča 360-stopinjske panoramske poglede lokacij na vseh sedmih celinah. Sklenili smo, da bomo rešitev iskali in razvijali postopoma. Najprej smo poiskali nekaj osnovnih informacij in primerov uporabe Google Street View API. Nato smo korak za korakom razvijali prototip prikaza nepremičnin. Google Street View ponuja panoramski pogled iz določenih cest v celotnem območju pokritosti. Pokritost Google Street View API-ja je enaka kot v aplikaciji Google Maps. Google Maps Javascript API zagotavlja brskalniku Street View servis za pridobivanje in obdelovanje panoramskih posnetkov, ki se uporabljajo v Google Maps Street View. Street View storitev je podprta v brskalniku.

Google Street View je lahko uporabljen v samostojnem DOM elementu. Najbolj pregleden način prikaza lokacije je prikaz na zemljevidu. Google Street View je privzeto omogočen na zemljevidu in nadzor Street View možiclja (ang. Pegman) je vključen v navigacijo (povečava in orientacija). Znotraj zemljevida je Street View mogoče tudi onemogočiti. V nastavitvah zemljevida nastavimo nastavitev *streetViewControl* na vrednost *false*. Spremenimo lahko tudi privzeti položaj na zemljevidu. Street View možicelj omogoča ogled panorame neposredno na zemljevidu. Ko kliknemo na možiclja in ga držimo, se zemljevid osveži, nekatere ulice se obarvajo modro. Možiclja postavimo in spustimo na modro obarvani ulici oz. cesti in prikaže se Street View panorama na označenem mestu. Street View slike so podprte z uporabo objekta *StreetViewPanorama*, ki ga zagotavlja API vmesnik za Street View pregledovalnik. Vsaka mapa vsebuje privzet Street View pogled (panoramo), ki ga je mogoče naložiti z metodo zemljevida *getStreetView*. Ko se doda nadzor na zemljevidu z nastavitvijo lastnosti zemljevida *streetViewControl* na vrednost *true*, se avtomatsko prikaže panorama uličnega pogleda. API omogoča, da sami nastavimo lokacijo panorame z nastavitvijo koordinat. S funkcijama *setPosition* in *setPov* nastavimo lokacijo in usmerjenost. Z drugo funkcijo spremenimo usmerjenost pogleda na lokaciji - točka pogleda (ang. Point-of-View) - POV. Street View določa umestitev poudarkov kamere za sliko, vendar ne opredeljuje orientacije fotoaparata za to sliko. Objekt *StreetViewPov* namreč omogoča dve lastnosti:

- *heading* definira zasuk okoli kamere v stopinjah od severa. Zasuk se meri v smeri urinega kazalca (90 stopinj je vzhod).
- *pitch* definira usmerjenost - kot fotoaparata gor ali dol. Koti se merijo kot pozitivne vrednosti stopinj za pogled navzgor in negativne vrednosti navzdol.

5.2.2 Prikaz uličnega pogleda iz najbližje lokacije do nepremičnine

V začetku smo iskali način, kako na podlagi koordinat sploh prikazati nepremičnino v uličnem pogledu. Na spletu smo poiskali primer [16], spremenili osnovne podatke in tako prikazali nepremičnino kot je prikazano v Primeru kode 5.3:

```
1  function initialize() {
2      var coordinates = { lat: 45.8375475, lng: 14.9177981 };
3      var map = new ↩
           google.maps.Map(document.getElementById("map"), {
4           center: coordinates, zoom: 1 });
5      var panorama = new google.maps.StreetViewPanorama(
6           document.getElementById("pano"), {
7               position: coordinates,
8               pov: { heading: -133, pitch: 0, zoom: 1 } });
9      map.setStreetView(panorama);
10 }
```

Listing 5.3: Funkcija primera Google Street View pogleda.

S prilagoditvijo primera smo prikazali nepremičnino na podlagi podanih koordinat in tako naredili prvi korak do rešitve prikaza nepremičnine z najbližje posnete točke do željene lokacije. Začetni primer smo nadgrajevali z dodatnimi funkcijami API-ja in v vsakem koraku našli dodaten izziv za izboljšanje prikaza. Problem pogleda z najbližje točke smo rešili s pomočjo servisa *StreetViewService* in s klicem funkcije servisa *getPanoramaByLocation*. Glede na podane parametre, vrne funkcija najbližjo zajeto sliko do podane lokacije. Podani parametri so koordinate lokacije in največja razdalja med točko zajemanja slike in lokacije. Na podlagi vrnjenega podatka o točki zajemanja slike in statusa smo razbrali, ali je bila zajeta slika pridobljena in kakšna je lokacija točke zajemanja. Iz podatka o statusu smo ugotovili, ali morda lokacije ni bilo mogoče najti oz. ulični pogled ni bil naj-

dem. V primerih, ko so lokacije zelo oddaljene od najbižje točke zajemanja, običajno nismo pridobili panoramske slike. Omenjena funkcija je kot izhodni podatek vrnila tudi podatke o lokaciji kamere, ki je zajemala panoramsko sliko.

5.2.3 Prikaz Street View pogleda usmerjen v smeri nepremičnine

Ko smo testirali prikazovanje nepremičnin, smo naleteli na težavo; ob kliku na gumb, uporabniku še vedno ni prikazana nepremičnina. Čeprav je bil možicelj že avtomatično postavljen v bližino nepremičnine, je bila vseeno potrebna ročna nastavitev orientacije levo-desno.

Res da je možicelj avtomatično postavljen v bližino nepremičnine, vendar je potrebno ročno nastaviti orientacijo levo-desno. Z raziskovanjem o API-ju smo našli podatek, da je to nastavitev *heading*. Glede na to, da je za uporabnika pomembno, da se ta podatek pogledu nastavi avtomatsko, smo se odločili, da najdemo ustrezno rešitev. Našli smo funkcijo, ki na podlagi podatkov o lokaciji zajemanja slike, izračuna zasuk levo-desno. Na podlagi podatka o lokaciji kamere in ciljni lokaciji, smo s funkcijo *computeHeading* pridobili podatek, kakšen je zasuk iz lokacije zajemanja panoramske slike v smer cilja - nepremičnine, ki smo jo želeli prikazati.

5.2.4 Dodajanje zastavic

V začetnih nastavitvah smo nastavili največjo oddaljenost nepremičnine od lokacije kamere na 100 metrov. V nekaterih primerih se je izkazalo, da nepremičnine zaradi velike oddaljenosti niti ne vidimo. Zgodi se lahko tudi, da je med lokacijo zajemanja slike in iskano nepremičnino vmes še ena nepremičnina. Tako uporabnik ne ve, ali je prikazana prava nepremičnina, ali je nepremičnina skrita nekje zadaj. Zato smo iskali rešitev, ki že v pogledu označi lokacijo nepremičnine. Na ulični pogled smo dodali na mesto nepremičnine zastavico (*marker*). Primer kode 5.4 za dodajanje zastavice:

```
1 new google.maps.Marker({
2     position: new google.maps.LatLng(45.8375475, ↩
          14.9177981),
3     map: new ↩
          google.maps.Map(document.getElementById("map"), ↩
          mapOptions),
4     title: "Ciljna lokacija"
5 });
```

Listing 5.4: Primer dodajanja zastavice, ki označuje nepremičnino.

Velikost zastavice se je premosorazmerno manjšala z večanjem oddaljenosti nepremičnine od točke zajemanja slike.

5.2.5 Iskanje najbolšega posnetka uličnega pogleda

Omenili smo že, da v primerih, ko ulični pogled ni mogoč, uporabniku le izpišemo obvestilo, da prikaz uličnega pogleda ni mogoč. Podobno smo naredili tudi v primerih, ko lokacija ni najdena. Takrat uporabniku izpišemo obvestilo, da naslova ni mogoče najti. Funkciji za iskanje panoramske slike smo nastavili vrednost največje oddaljenosti od lokacije snemanja do nepremičnine. Na podlagi preučitve delovanja funkcije *getPanoramaByLocation* smo razvili rešitev postopnega pridobivanja uličnega pogleda. Za iskanje najboljšega uličnega pogleda smo uporabili rekurzivni klic funkcije, ki išče lokacijo oz. sliko panorame na podlagi podanih koordinat nepremičnine in največje oddaljenosti nepremičnine. V vsakem koraku smo največjo oddaljenost povečali za nekaj metrov in tako pridobili najbolj ustrezen ulični pogled na nepremičnino. Za pohitritev postopka smo uporabili funkcijo *computeDstanceBetween*, ki na podlagi točke snemanja in ciljne lokacije izračuna oddaljenost. S pomočjo omenjene funkcije smo dobili natančen podatek, koliko je neka nepremičnina oddaljena od točke zajemanja panorame. Primer izpisa oddaljenosti nepremičnine na Sliki 5.5.

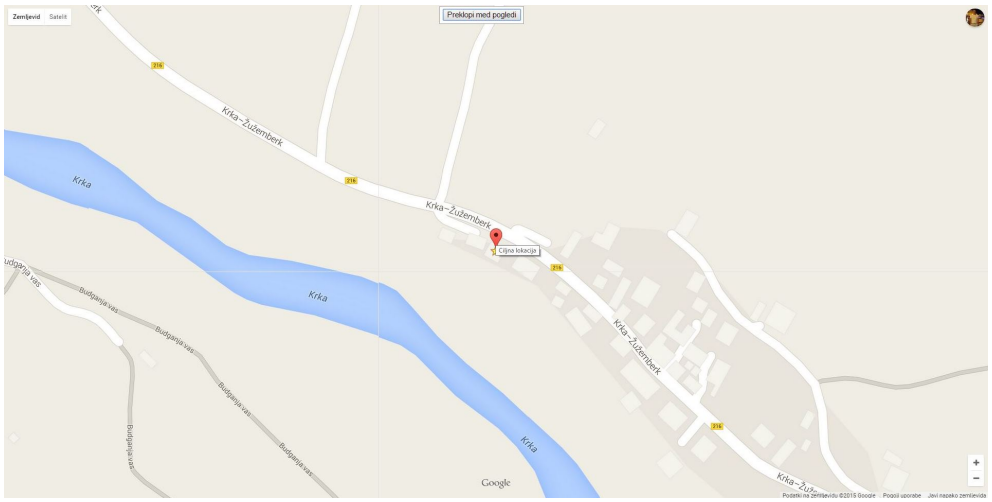
S testiranjem prototipa rešitve smo ugotovili, da je potrebno postaviti

Slika 5.5: Prikaz oddaljenosti nepremičnine od točke zajemanja panorame.

zgornjo mejo oddaljenosti nepremičnine. V primeru, da je največja oddaljenost s funkcijo dosežena, uporabniku sporočimo, da ulični pogled ni mogoč. V primerih, ko je nepremičnina oddaljena več 10m, se je izkazalo, da pogled nepremičnine ni več pregleden.

5.2.6 Prikaz nepremičnine na zemljevidu

V primerih, ko ulični pogled ni mogoč ali pa lokacije nepremičnine ni mogoče najti, prototip funkcionalnosti prikaza nepremičnine z uličnim pogledom uporabniku izpiše obvestilo, da pogled ni mogoč. V takem primeru je za uporabnika pomembno, da se prikaže alternativa uličnemu pogledu. Predvideli smo, da bi v takšnih primerih uporabniku ustrezal vsaj prikaz nepremičnine na zemljevidu. Zato smo se odločili, da uporabniku v vsakem primeru omogočimo prikaz tudi na zemljevidu. V primerih, ko ulični pogled ni mogoč, je viden le prikaz na zemljevidu, sicer pa oba. Za lažjo orientacijo smo nepremičnino na zemljevidu označili s posebno zastavico. Med pogledoma je mogoče preklapljati z gumbom na vrhu prikaza kot je prikazano na Sliki 5.6.



Slika 5.6: Prikaz nepremičnine na zemljevidu označene z zastavico in možnost preklapljanja med prikazoma.

5.2.7 Problem neposredne bližine nepremičnine

V nekaterih primerih testiranja smo odkrili težavo prikaza nepremičnine in sicer, npr. če je nepremičnina v neposredni bližini točke fotografiranja. S spremembo orientacije levo-desno, gor-dol in spremembo lokacije prikaza pogleda smo lahko našli iskano nepremičnino. Ugotovili smo, da je v takih primerih potrebno ustrezno nastaviti nastavitve Street View *pitch*. Določili smo, da se za poglede, ki prikazujejo nepremičnine, oddaljene od točke snemanja manj kot 5 metrov, avtomatsko nastavi nastavitev usmerjenosti gor-dol. V takšnih primerih smo usmerili pogled navzdol.

5.3 Avtomatsko pridobivanje podatkov prek spletnega servisa

5.3.1 Definiranje cilja

V Poglavju 3 smo pogledali podroben opis obstoječega načina pridobivanja ETN podatkov. Tak način uporabnikom aplikacije onemogoča operiranje s svežimi podatki. Zato smo se odločili, da izdelamo prototip funkcionalnosti nadgradnje aplikacije, ki avtomatično prenese podatke prek GURS spletnega servisa v podatkovno bazo aplikacije. Na začetku razvoja smo postavili cilj, da razvijemo rešitev, ki se vsakodnevno avtomatsko proži in prek spletnega servisa prenese podatke v našo podatkovno bazo. "Skozi postopek pridobivanja podatkov naj se ustvarja poročilo o napredovanju in se ob koncu pošlje skrbnikom sistema." To je za nas predstavljalo najbolj optimalno rešitev.

5.3.2 Potek komunikacije s servisom

Pred pričetkom razvoja smo se posvetili analizi, kako deluje GURS spletni servis. Iz specifikacije poteka komunikacije smo ugotovili, da se morajo odjemelci spletnega servisa predhodno registrirati ter pridobiti pravico za dostop do posameznih funkcionalnosti servisa. Identifikacija uporabnika se izvaja na podlagi spletnega digitalnega potrdila uporabnika. Komunikacija in prenos podatkov prek servisa pa potekata v več korakih.

5.3.3 Dostop z uporabo HTTP GET zahtev

Do spletnega servisa je mogoče dostopati po dveh poteh:

- dostop z uporabo HTTP GET zahtev,
- dostop z uporabo SOAP.

Zaradi preprostosti uporabe, smo se odločili za dostop z uporabo HTTP GET zahtev [10]. HTTP protokol [3] omogoča komunikacijo med odjemalci

in strežniki. Deluje kot zahteva-odgovor med odjemalcem in strežnikom. GET je vrsta zahteve protokola HTTP, ki deluje kot zahteva za podatke iz določenega vira. Poizvedba (par ime=vrednost) se pošlje kot zahteva v URL-ju zahteve GET.

5.3.4 Koraki za pridobitev podatkov

Prvi korak - pridobitev žetona

Prvi korak v komunikaciji je pridobitev sejnega žetona (*sessionId*). Za pridobitev žetona smo v začetku poskusili z dostopom do servisa prek znanega *URL*-ja s spletnim brskalnikom. Ob dostopu do strani smo izbrali testni certifikat in dobili odgovor:

```
1   −1, Napaka −4: Uporabnisko ime in geslo sta enaka!
```

Listing 5.5: Obvestilo servisa o napaki - nespremenjeno geslo.

Zaradi napake smo se obrnili na GURS. Dobili smo odgovor, da je potrebno nastaviti svoje geslo. Ob naslednjem poskusu pridobitve žetona smo naleteli na novo napako:

```
1   −1, Napaka −1: Uporabnik CENILCI nima ustreznih ↩
      pravic do aplikacije "OWS"!
```

Listing 5.6: Obvestilo servisa o napaki - ni dodeljenih pravic.

Po obvestilu servisa o novi napaki, smo se ponovno obrnili na GRUS. Težava je bila, da za testni certifikat ni bilo dodeljenih pravic dostopa do servisa. Ko so bile pravice za dostop dodeljene, smo v novem poskusu uspeli pridobiti žeton, ki ga nujno potrebujemo, preden pričnemo s klici ostalih funkcionalnosti servisa.

```
1   BAU71Y726JQSMPPK0HRW92RSGKTUHU
```

Listing 5.7: Pridobljen žeton.

Zaradi potrebe pridobivanja žetona v aplikaciji, smo implementirali funkcijo, ki pridobi žeton s HTTP GET klicem servisa. Aplikacija priloži zahtevku testni certifikat, s katerim se predstavi servisu. Servis zahtevek obdela in aplikaciji vrne žeton. Omejitev vsakega sejnega žetona je 30 minut. V primeru, da je izdelava odgovora na zahtevo dolgotrajna, je sejni žeton potrebno osvežiti ali zahtevati novega. S klicem funkcije *getSessionId* smo pridobili sejni žeton, katerega smo uporabljali v naslednjih klicih funkcij servisa.

Drugi korak - preštevanje poslov

Pred izvozom podatkov ETN prek spletnega servisa je smiselno, da posle preštejemo. Prenos poslov je mogoče s kriteriji tudi omejiti, z namenom, da prenesemo le tiste posle, ki jih želimo. Za potrebe vsakodnevnega prenosa je potrebno prenesti vse posle, ki so se v register vpisali prejšnji dan. Za omejitev prenosa smo uporabili parametra *datumUveljavitveOd* in *datumUveljavitveDo*. Zaradi poenostavitve prenosa podatkov, smo uporabili še parameter *vrstaPosla*. Ker se struktura podatkov za kupoprodajne in najemne posle razlikuje, smo vsako vrsto poslov prenašali posebej. Za preštevanje poslov smo usvarili novo HTTP GET zahtevo. Zahtevo smo izvedli s klicem URL naslova servisa in mu dodali naslednje parametre:

- ime klica funkcije za preštevanje poslov - *prestejPosle*,
- sejni žeton - *sessionId*,
- vrsta posla - *vrstaPosla*,
- datum uveljavitve od - *datumUveljavitveOd*,
- datum uveljavitve do - *datumUveljavitveDo*.

Servis je vrnil odgovor v obliki XML dokumenta, iz katerega je bilo mogoče razbrati:

- število ustreznih poslov, če so iskalni pogoji veljavni in koliko poslov obstaja glede na kriterije,

- da ne obstaja noben posel glede na podane pogoje,
- da je prišlo do napake, če iskalni pogoji ne zadoščajo zahtevam glede formata, vrednosti ali če so med seboj neusklajeni.

V primeru uspešnega odgovora, smo dobili podatek, koliko poslov je na voljo za prenos. Če je bilo število poslov pozitivno, smo nadaljevali s postopkom izvoza/prenosa podatkov, sicer smo prenehali izvajati proces pridobivanja podatkov za nastavljeno iskanje.

Tretji korak - zahtevaj posle

Po prejemu podatka o številu poslov za podane kriterije, smo pričeli z razvojem funkcionalnosti za zahtevo po izvozu poslov. Izvoz poslov poteka v dveh stopnjah. Prva stopnja je zahteva po izvozu poslov. Tudi ob zahtevi poslov smo uporabili enake parametre kot pri preštevanju poslov, le da smo v tej fazi klicali funcijo servisa *zahtevajPosle*. Pri zahtevi po izvozu poslov smo uporabili še dva dodatna pogoja, in sicer odmik ter mejo. Dodatna pogoja sta namenjena omejitvi množice poslov in sta povezana z zaporednimi številkami poslov v seznamu za izvoz. Parameter *odmik* pomeni, koliko poslov z začetka seznama naj servis izpusti pri vračanju rezultatov. Drugi parameter *meja* pa pomeni, največ koliko poslov naj se izvozi s seznama. Posle smo izvažali po delih, zato ni bilo potrebno predolgo čakati na odgovor servisa za posle. Za izvažanje poslov po delih smo uporabili omenjena dodatna parametra. Odgovor na zahtevo po izvozu poslov je vseboval:

- podatek o številu ustreznih poslov, če so iskalni pogoji veljavni,
- podatek, če ne obstaja noben posel glede na iskalne pogoje,
- podatek o napaki, če iskalni pogoji ne zadoščajo zahtevam glede formata, vrednosti ali so med seboj neusklajeni,
- podatek *oznakaZahtevka*, ki smo ga uporabili v naslednji stopnji - prevzem poslov.

Četrti korak - prevzemi posle

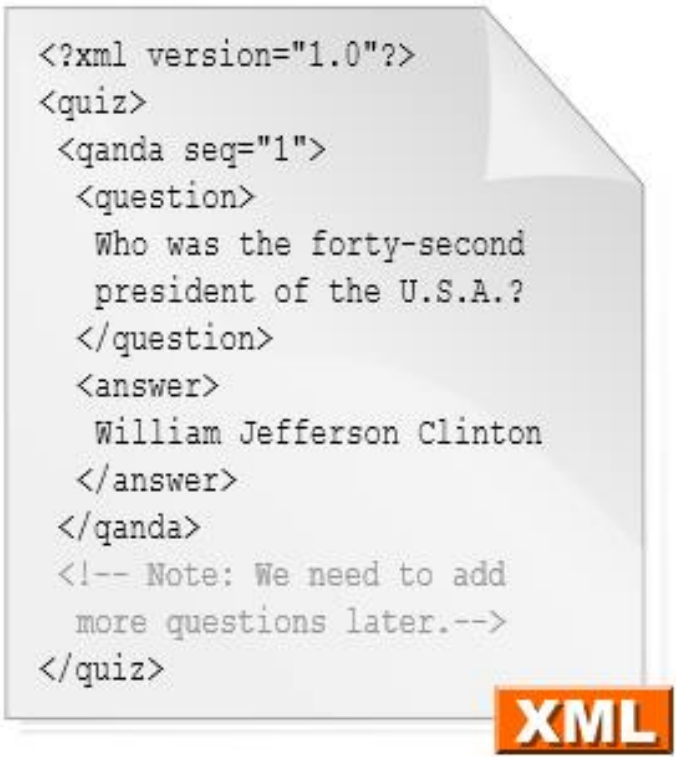
Druga stopnja izvoza podatkov se zaključi z zahtevo po prevzemu poslov. To je tudi zadnji korak prenosa podatkov prek servisa. Posle prevzamemo s klicem servisa in podanim parametrom za klic funkcije *prevzemiPosle*. H klicu dodamo še sejni žeton in številko *oznakaZahtevka*, ki smo jo pridobili v prejšnjem koraku zahteve poslov. Odgovor na zahtevo po prevzemu poslov lahko vsebuje naslednje podatke:

- podatek o statusu zahteve - koliko poslov je že obdelanih in koliko jih je še ostalo,
- seznam poslov, v kolikor je seznam že pripravljen,
- podatke o napaki, če je do nje prišlo.

Zaradi priprave podatkov na prevzem podatkov, je bilo potrebno čakanje na odgovor servisa. V zanki smo vsake 3 sekunde preverili, ali so podatki že pripravljeni. Čakanje smo izvedli s klicem *Thread.Sleep(3000)*. Prevzeti podatki so bili zapisani v obliki XML. Pridobljene podatke je bilo potrebno nato še obdelati, več o obdelavi v nadaljevanju. Po zaključeni obdelavi prevzetih podatkov, smo jih zapisali v podatkovno bazo. Šele nato smo lahko zahtevali prevzem naslednjega dela poslov, v kolikor še nismo prenesli vseh. Zahtevo po ostalih poslih smo izvedli podobno kot prej, le da smo povečali vrednost odmika.

5.3.5 XML

XML [2] je preprost računalniški jezik podoben jeziku HTML. Omogoča format za opisovanje strukturiranh podatkov. Struktura jezika je zelo preprosta in enostavna za razumevanje. Uporablja se za komunikacije. Jezik vsebuje značke (ang. tag) [18], vsaka izmed njih pa ima običajno svojo vrednost, kot je razvidno na Sliki 5.7.



Slika 5.7: Primer XML zapisa.

5.3.6 Razčlenjevalniki

Prevzeti podatki o poslih so imeli obliko zapisa XML. Zato je bilo potrebno razviti postopek oz. funkcijo, ki je podatke iz XML oblike zapisa, spremenila v obliko podatkov primernih za zapis v podatkovno bazo. Vsako prebrano vrednost iz XML smo shranili v instanco ustreznega razreda in nato instanco zapisali v podatkovno bazo. Zaradi različnih shem podatkov za posle, dele poslov in šifrante, je bilo potrebno napisati šest (6) različnih funkcij, t.i. razčlenjevalnikov. Vsi prejeti podatki so bili podatkovnega tipa *besedilo*. V nekaterih primerih je bil tak tip sprejemljiv, v drugih pa je bilo potrebno podatke pretvoriti v drugačen podatkovni tip, npr. celo število, decimalno število, datum. Glavna težava v tem koraku je bila, ker nismo poznali sheme zapisa podatkov. Tako smo iz odgovorov servisa sproti ugotavljali, kako so

posamezne vrednosti podatkov poimenovane oz. označene. V nekaterih odgovorih so se pojavili tudi podatki, ki jih nismo potrebovali. Takšne podatke smo obravnavali kot neuporabne.

5.3.7 Potek žetona

Vsak izveden klic po zahtevi smo opremili s sejnim žetonom, ki je veljaven 30 minut. Pričakovali smo, da servis lahko postane neodziven ali pa priprava odgovorov lahko traja dlje. V primeru, ko poteče veljavnost žetona, je potrebno žeton podaljšati oz. pridobiti novega. Klici za preštevanje in zahtevanje poslov so se običajno izvedli zelo hitro, kar je pomenilo, da ta dva klica ne predstavljata ozkega grla. Največ časa za izvedbo je potreboval klic funkcije servisa 'prevzemi posle'. Ta funkcija je bila počasna zato ker je ob klicu pripravljala podatke za prenos. To pa je časovno potratno za servis. Zato smo pred vsakim klicem za prevzem poslov preverili, koliko časa je še ostalo za prevzem poslov. Odločili smo se, da v primerih, ko je zadnji pridobljeni sejni žeton starejši od 29 minut in pol, zahtevamo novega. Servis omogoča, da se sejni žeton ob prevzemu poslov ralikuje od žetona, ki smo ga uporabljali za zahtevo poslov.

5.3.8 Poročilo poteka komunikacije

Zaradi odvisnosti funcionalnosti od spletnega servisa GURS za prenos podatkov ETN, smo na nekaterih mestih programa predvidevali, da lahko pride do napake. Napake so lahko različne, npr.: neodzivnost servisa, nepričakovana vrednost spremenljivke ipd. Zaradi nadzora poteka prenosa podatkov in kontrole pravilnega delovanja prototipa funkcionalnosti, smo se odločili beležiti stanje programa na različnih mestih. Implementirali smo funkcionalnost, ki shranjuje zapise o napredku, v začasno besedilno datoteko skupaj s časovno oznako. Med drugimi smo beležili tudi: da se je pridobil sejni žeton, da je žeton potekel, da smo pridobili odgovor, koliko poslov je na voljo za prenos, koliko poslov smo uspešno vpisali v podatkovno bazo, kater podatek v XML

zapisu nismo shranili oz. obravnavali itd. Ob koncu postopka prenosa podatkov, smo poslali skrbnikom sistema poročilo o postopku prenosa. Začasno datoteko, v katero smo shranjevali zapise o napredku postopka prenosa podatkov, smo kot priponko poslali v email sporočilu.

5.3.9 Optimizacija

S testiranjem prototipa funkcionalnosti smo v programu odkrili dele kode, kjer so možne optimizacije. Ena izmed vidnejših optimizacij je uporaba niti. Čas, ko čakamo servis za prevzem poslov, lahko rečemo, da je izgubljen. Vsak prevzem poslov bi tako predali niti, program pa bi lahko takoj zahteval nov del podatkov. Na ta način bi se hitreje končal glavni proces pridobivanja podatkov. Obstaja tudi verjetnost, da s takim načinom ne bi pohitrili procesa pridobivanja, saj bi s tem zahtevali več hkratnih operacij spletnega servisa. V primeru, da ima spletni servis omejitev kapacitet na vsakega uporabnika, z uporabo niti ne bi uspeli pohitriti programa za prenos podatkov.

5.3.10 Konzolna aplikacija, ki proži proces prenosa podatkov

Razvito funkcionalnost je mogoče uporabiti kot knjižnico. Za proženje funkcij prototipa funkcionalnosti za prenos podatkov prek GURS spletnega servisa, smo napisali preprost program - konzolno aplikacijo. Apikacijo smo razvili s pomočjo programa Visual Studio. Prevedeno aplikacijo je bilo mogoče zagnati le s klikom na datoteko tipa *exe* in tako sprožiti postopek prenosa podatkov za dan predvčerajšnjim. Za prenos dva dni starih podatkov smo se odločili iz razloga, da GURS po vnosu podatke še nekaj časa obdeluje, preden so dostopni. Zato prenos poslov iz prejšnjega dne ni smiselen, saj posli najverjetneje ne bodo pripravljeni za prenos. Podatki so se prenesli preko servisa in zapisali v podatkovno bazo. Glede na to, da je potrebno za prenos vsak dan ročno zagnati program, smo se odločili najti način, da bo zaganjanje programa samodejno. Na email smo želeli prejeti poročilo o

prenosu podatkov.

5.3.11 Uporaba programa 'Task Scheduler'

Za avtomatsko proženje programa smo uporabili program *Task Scheduler*. V programu smo dodali novo opravilo. Nastavili smo, da opravilo zažene konzolno aplikacijo, ki prenese podatke prek servisa. Opravilu smo nastavili, da se avtomatsko izvede vsak dan ob istem času. Po uspešni nastavitvi smo opravilo shranili in ga zagnali. Izvajanje smo spremljali nekaj dni, prav tako poročila izvajanja prototipa funkcionalnosti, ki prenaša podatke. Na ta način smo ugotovili, ali prihaja do kakršnihkoli napak.

5.3.12 Pridobitev podatkov za obdobje od zadnjega uvoza do začetka avtomatskega prenosa podatkov

Po zaključenem razvoju prototipa funkcionalnosti samodejnega pridobivanja ETN podatkov prek GURS spletnega servisa, smo odkrili še neobravnavan problem. Od zadnje pošiljke ETN podatkov v obliki datotek do trenutka, ko smo zaključili z razvojem prototipa za avtomatsko pridobivanje podatkov, je preteklo kar nekaj časa. V tem času so se v ETN dodali novi posli, ki jih na implementiran način ne bo mogoče pridobiti, saj se prenašajo le dva dni stari podatki. Zato smo razvito konzolno aplikacijo nadgradili in implementirali postopek, tako da za vsak manjkajoči dan posebej pridobi podatke preko spletnega servisa. Na tak način smo prenesli manjkajoče podatke o poslih.

Poglavje 6

Sklepne ugotovitve

V okviru diplomskega dela je bil razvit prototip prilagoditve aplikacije na novo shemo podatkov, prototip prikaza nepremičnin s pomočjo Google Street View in prototip funkcionalnosti avtomatskega prenosa podatkov prek GURS spletnega servisa v podatkovno bazo aplikacije. Iz testov je razvidno, da smo dosegli vse zastavljene cilje iz Poglavja 3. Skozi celoten razvoj smo stremeli k izboljšanju uporabniške izkušnje. To smo uspeli zagotovili tako, da smo operacije sprogramirali optimalno, kar se odraža na hitri odzivnosti aplikacije. Procese smo minimalizirali in jih naredili enostavne za uporabo, tako da uporabnik do rešitve pride v le nekaj korakih. Avtomatizirali smo tudi nekatere dele procesov, da ni potrebno konstantno ponavljanje postopkov.

V začetku smo se osredotočili na prilagoditev obstoječe aplikacije na novo shemo podatkov. Z natančno analizo razlik med obstoječo in novo shemo podatkov smo lahko natančno ocenili, kje je potrebno narediti največ sprememb. S prilagoditvijo smo nadgradili aplikacijo do faze, ki je sposobna obdelovati podatke po novi shemi. Aplikacija je postala bolj uporabna, saj prototip podpira tudi obdelavo najemnih poslov.

V drugem delu smo iskali najboljšo rešitev prikaza nepremičnin. Za prikaz smo uporabili Google Street View API. Na ta način smo uspeli nepremičnino na podlagi koordinat, prikazati v panoramskem pogledu. Ključen napredek tega dela razvoja je, da lahko uporabnik z enim klikom pridobi prikazano

nepremičnino kot ulični pogled. Prototip prikaza nepremičnin poleg prikaza v uličnem pogledu, omogoča tudi prikaz lege nepremičnine na zemljevidu. Z različnimi dodanimi obvestili, so uporabniku na voljo tudi dodatne informacije. Ta del je uporaben tudi za prikaz nepremičnin v katerikoli drugi aplikaciji. Prototip je zelo preprost za integracijo v drug sistem, saj deluje kot samostojen modul. Edina omejitev pri tej funkcionalnosti je, da deluje le na območjih, kjer je Street View mogoč. Za to funkcionalnost bi bilo mogoče pohitriti iskanje lokacije s spremembo trenutne logike. Preprostejša optimizacija bi bila implementacija logike nastavljanja povečave in orientacije gor-dol. Izboljšavo za povečavo bi lahko naredili na podlagi oddaljenosti nepremičnine.

V zaključnem delu smo se posvetili avtomatizaciji obstoječega načina pridobivanja podatkov ETN. Implementirali smo rešitev, ki z dnevnimi klici spletnega servisa avtomatsko pridobi podatke in jih zapiše v podatkovno bazo. Tak način pridobivanja podatkov uporabnikom omogoča operiranje s svežimi podatki. To vpliva na dobro uporabniško izkušnjo. Prototip funkcionalnosti je mogoče uporabiti v drugih sistemih, ki želijo avtomatsko pridobivati podatke prek spletnega servisa z GURS-a. Z manjšo predelavo programske kode prototipa, je mogoče funkcionalnost uporabiti tudi v drugih sistemih za pridobivanje podatkov prek servisa. Z implementacijo te rešitve smo dosegli zastavljeni cilj - avtomatsko pridobivanje podatkov ETN prek GURS spletnega servisa in pošiljanje poročila na email. Optimizacija pri prototipu prenosa podatkov bi bila, v prvi meri, že prej omenjena uporaba niti. Na nekaterih delih je mogoče zaslediti podvajanje programske kode, kar bi lahko optimiziralo delovanje funkcionalnosti.

Kljub možnim izboljšavam prototipov je mogoče trditi, da smo zadostili vsem ciljem naloge. Iz rezultatov testiranja je razvidno, da se je uporabniška izkušnja aplikacije izboljšala, del procesov je enostavnejši in popolnoma avtomatiziran.

Literatura

[1] M. Chand, "Programming C# for Beginners", 2014

[2] M. Chand, "Programming XML with C#", 2011

[3] B. S. Davie, L. L. Peterson, "Computer Networks, Fifth Edition: A Systems Approach 5th", 2011

[4] A. I. Din, "Structured Query Language (SQL) - A Practical Introduction", 1994

[5] M. MacDonald, "Beginning ASP.NET 4.5 in C#", 2012

[6] J. Smith, "Build and Design A Website: An Instruction to HTML and CSS", 2013

[7] ASP.NET [Online]. Dosegljivo: https://en.wikipedia.org/wiki/ASP.NET. [Dostopano 13. 8. 2015].

[8] C Sharp (programming language) [Online]. Dosegljivo: https://en.wikipedia.org/wiki/C_Sharp_(programming_language). [Dostopano 13. 8. 2015].

[9] HTML [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/HTML. [Dostopano 13. 8. 2015].

[10] HTTP Methods: GET vs. POST [Online]. Dosegljivo: http://www.w3schools.com/tags/ref_httpmethods.asp. [Dostopano 27. 8. 2015].

[11] JavaScript [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/JavaScript. [Dostopano 13. 8. 2015].

[12] SQL [Online]. Dosegljivo: https://en.wikipedia.org/wiki/SQL. [Dostopano 13. 8. 2015].

[13] SQL [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/SQL. [Dostopano 13. 8. 2015].

[14] Stored Procedures (Database Engine) [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/library/ms190782.aspx. [Dostopano 12. 8. 2015].

[15] Street View: kako najti fotografijo svoje ulice in hiše. [Online]. Dosegljivo: http://www.siol.net/novice/tehnologija/racunalnistvo/2014/01/street_view.aspx. [Dostopano 6. 8. 2015].

[16] Street View Service [Online]. Dosegljivo: https://developers.google.com/maps/documentation/javascript/examples/streetview-simple. [Dostopano 17. 8. 2015].

[17] Street View Service [Online]. Dosegljivo: https://developers.google.com/maps/documentation/javascript/streetview. [Dostopano 14. 8. 2015].

[18] XML [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/XML. [Dostopano 27. 8. 2015].